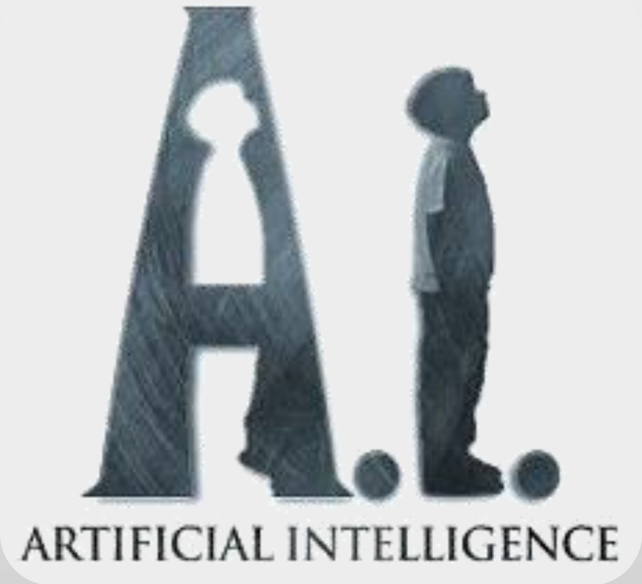


First-Order Logic

Chapter 8

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
INSTITUTIONAL OFFICE

Announcements

- Semester Test 2
  - Thursday, 11 November 2021
  - Everything up till then
  - (mostly the stuff after the Semester Test 1)
- Remember the peer assessment on Practical Assignment 5!!

Overview of lecture

- The representation of simple domains
  - The theory of natural numbers
  - Sets
  - The wumpus world
- Knowledge engineering in first-order logic

Theory of natural numbers

- Need
  - Predicate: NatNum
  - Constant symbol: 0
  - Function symbol: S
- Recursive definition

  $NatNum(0)$

  $\forall\, n\ NatNum(n) \Rightarrow NatNum(S(n))$
- Examples

  $0, S(0), S(S(0)), S(S(S(0)))$, etc.

Theory of natural numbers

- Constrain successor function

  $\forall\, n\ 0 \neq S(n)$

  $\forall\, m,n\ m \neq n \Rightarrow S(m) \neq S(n)$

Theory of natural numbers

- Addition

  Binary function

  $\forall$ m NatNum(m) $\Rightarrow$ +(0, m) = m

Theory of natural numbers

- Addition
  - $\forall$ m NatNum(m) $\Rightarrow$ +(0, m) = m
  - $\forall$ m,n NatNum(m) $\wedge$ NatNum(n) $\Rightarrow$ +(S(m), n) = S(+(m,n))

Theory of natural numbers

- Addition

  $\forall$ m NatNum(m) $\Rightarrow$ +(0, m) = m

  $\forall$ m,n NatNum(m) $\land$ NatNum(n) $\Rightarrow$ +(S(m), n) = S(+(m,n))

  $\forall$ m,n NatNum(m) $\land$ NatNum(n) $\Rightarrow$ (m + 1) + n = (m + n) + 1

Theory of natural numbers

- Addition

  $\forall$ m NatNum(m) $\Rightarrow$ +(0, m) = m

  $\forall$ m,n NatNum(m) $\wedge$ NatNum(n) $\Rightarrow$ +(S(m), n) = S(+(m,n))

  $\forall$ m,n NatNum(m) $\wedge$ NatNum(n) $\Rightarrow$ (m + 1) + n = (m + n) + 1

- syntactic sugar: an extension to or abbreviation of the standard syntax that does not change the semantics

Theory of natural numbers

- Multiplication
  - Repeated addition
- Exponentiation
  - Repeated multiplication
- Integer division, remainders, prime numbers, etc.

Theory of natural numbers

- Multiplication
  - Repeated addition
- Exponentiation
  - Repeated multiplication
- Integer division, remainders, prime numbers, etc.
- Thus, the whole of number theory (including cryptography) can be built up from one constant, one function, one predicate and four axioms.

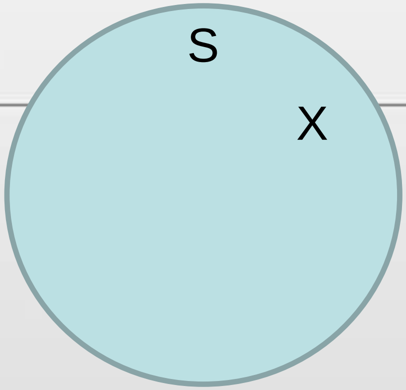
Sets

- The domain of sets is also fundamental to mathematics as well as to common sense reasoning.
- We want to be able to represent individual sets, including the empty set.
- We need a way to build up sets from elements or from operations on other sets.
- We will want to know whether an element is a member of a set
- We will want to distinguish sets from objects that are not sets.

Sets

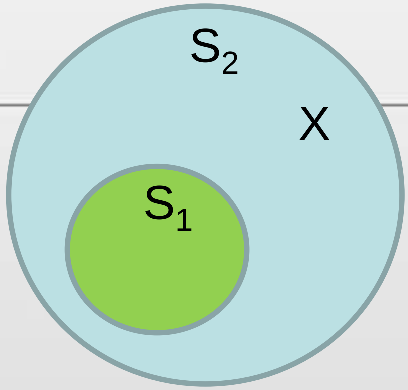
- Empty set: { }
- Predicate: Set
- Nothing is in the empty set, and the Set predicate is true for sets.

Sets



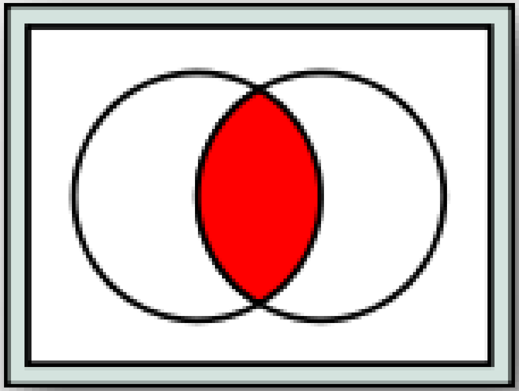
- Empty set: { }
- Predicate: Set
- Binary predicate: x ∈ s
- True if object x is in set S

Sets



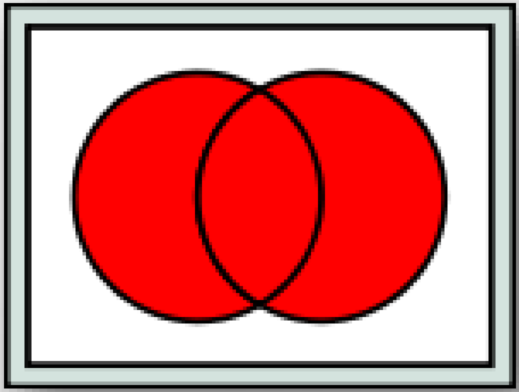
- Empty set: { }
- Predicate: Set
- Binary predicate: $x \in s$
- Binary predicate: $s_1 \subseteq s_2$
- S_1 is a subset of S_2
- or S_1 is the same as S_2

Sets



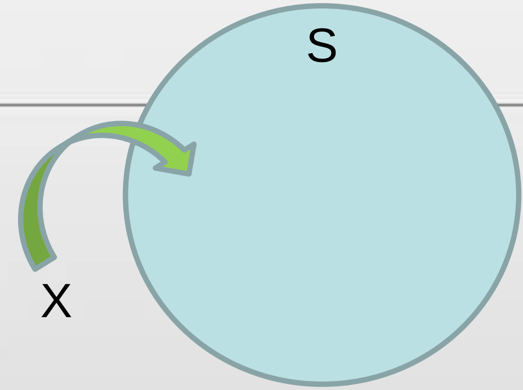
- Empty set: { }
- Predicate: Set
- Binary predicate: $x \in s$
- Binary predicate: $s_1 \subseteq s_2$
- Binary function: $s_1 \cap s_2$
- The returned set is the set of elements that are in both S_1 and S_2
- Note, a “binary function” refers to the number of arguments (2)

Sets



- Empty set: { }
- Predicate: Set
- Binary predicate: $x \in s$
- Binary predicate: $s_1 \subseteq s_2$
- Binary function: $s_1 \cap s_2$
- Binary function: $s_1 \cup s_2$
- The combination of the elements of S_1 and S_2 as a set

Sets



- Empty set: { }
- Predicate: Set
- Binary predicate: $x \in s$
- Binary predicate: $s_1 \subseteq s_2$
- Binary function: $s_1 \cap s_2$
- Binary function: $s_1 \cup s_2$
- Binary function: Add(x,s)
- The set that results from adding element x to set S

Sets

- Next we want to define the axioms that describe the world. Those things that need to be true for the world to make sense. There are 8 axioms we need to describe the world.

Sets

1. $\forall s\ Set(s) \Leftrightarrow (s = \{\ \}) \vee (\exists\ x,s_2\ Set(s_2) \wedge s = Add(x,s_2))$

- The only sets are the empty set, and the set that results from adding an element to a set.

Sets

1. $\forall s \; Set(s) \Leftrightarrow (s = \{\ \}) \vee (\exists x, s_2 \; Set(s_2) \wedge s = Add(x, s_2))$

- The only sets are the empty set, and the set that results from adding an element to a set.
- For all s, s is a set if and only if …

Sets

1. $\forall s\ Set(s) \Leftrightarrow (s = \{\ \}) \vee (\exists\ x,s_2\ Set(s_2) \wedge s = Add(x,s_2))$

- The only sets are the empty set, and the set that results from adding an element to a set.
- For all s, s is a set if and only if …
- s is the empty set OR …

Sets

1. $\forall s\ Set(s) \Leftrightarrow (s = \{\ \}) \vee (\exists\ x, s_2\ Set(s_2) \wedge s = Add(x, s_2))$

- The only sets are the empty set, and the set that results from adding an element to a set.
- For all s, s is a set if and only if …
- s is the empty set OR …
- There exists x and s_2 such that s_2 is a set AND …

Sets

1. $\forall s\ Set(s) \Leftrightarrow (s = \{\ \}) \vee (\exists\ x,s_2\ Set(s_2) \wedge s = Add(x,s_2))$

- The only sets are the empty set, and the set that results from adding an element to a set.
- For all s, s is a set if and only if …
- s is the empty set OR …
- There exists x and s_2 such that s_2 is a set AND …
- s is the same as x added to s_2.

Sets

1. $\forall s\ Set(s) \Leftrightarrow (s = \{\ \}) \vee (\exists\ x,s_2\ Set(s_2) \wedge s = Add(x,s_2))$
2. $\neg\exists\ x,s\ Add(x,s) = \{\ \}$

- There exists no set that when you add something to it it becomes the empty set.

Sets

1. $\forall s \; Set(s) \Leftrightarrow (s = \{\ \}) \vee (\exists\, x,s_2 \; Set(s_2) \wedge s = Add(x,s_2))$
2. $\neg\exists\, x,s \; Add(x,s) = \{\ \}$
3. $\forall\, x,s \; x \in s \Leftrightarrow s = Add(x,s)$

- Adding an element that exists in a set already has no effect. x is an element of s only if s is the same as s+x.

Sets

1. $\forall s\ Set(s) \Leftrightarrow (s = \{\ \}) \lor (\exists\ x,s_2\ Set(s_2) \land s = Add(x,s_2))$
2. $\neg\exists\ x,s\ Add(x,s) = \{\ \}$
3. $\forall\ x,s\ x \in s \Leftrightarrow s = Add(x,s)$
4. $\forall\ x,s\ x \in s \Leftrightarrow [\exists\ y,s_2\ (s = Add(y,s_2) \land (x = y \lor x \in s_2))]$

- The only members of a set are those elements that have been added to it.
- This is defined recursively saying that x is a member of s if and only if s is equal to some element y added to some set s_2, where either y is the same as x or x is a member of s_2.

Sets

5. $\forall\ s_1, s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$

- A set s_1 is a subset of another set s_2 if and only if all of the members of s_1 are members of s_2.

Sets

5. $\forall\ s_1, s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$
6. $\forall\ s_1, s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$

- sets s_1 and s_2 are equal only if s_1 is a subset of s_2 and s_2 is a subset of s_1 (the equality here is superfluous, but we defined it that way).

Sets

5. $\forall\ s_1,s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$
6. $\forall\ s_1,s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
7. $\forall\ x,s_1,s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$

- An object x is in the intersection of sets s_1 and s_2 if and only if it is an element of both sets.

Sets

5. $\forall\ s_1,s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$
6. $\forall\ s_1,s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
7. $\forall\ x,s_1,s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
8. $\forall\ x,s_1,s_2\ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

- An object x is in the union of two sets s_1 and s_2 if and only if it is an element of s_1 or s_2 or both.

Sets

5. $\forall\ s_1, s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$
6. $\forall\ s_1, s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
7. $\forall\ x, s_1, s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
8. $\forall\ x, s_1, s_2\ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

- With the constant empty set, the unary predicate set(s), the 2 binary predicates, the 3 binary functions and the 8 axioms we have described set theory!

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
INSTITUTIONAL OFFICE

The wumpus world

- Next we can describe the Wumpus World in first order logic.
- We discussed the propositional logic representation of the Wumpus World previously, but now we will see that the first order logic version is much more concise.

The wumpus world

- Expressions in first-order logic very compact.
- Remember that the agent receives 5 constants in a precept that indicates the time step: stench, breeze, glitter, bump, scream, time.
- An example of a precept sequence would be:

  Percept([Stench, Breeze, Glitter, None, None], 5)

The wumpus world

- Actions that can be performed:

  Turn(Right), Turn(Left), Forward, Shoot, Grab, Release, Climb

The wumpus world

- Actions that can be performed:

  Turn(Right), Turn(Left), Forward, Shoot, Grab, Release, Climb

- Best action is derived from the KB using the ASKVARS query.

  ASKVARS($\exists$a BestAction(a, 5))

The wumpus world

- Actions that can be performed:

  Turn(Right), Turn(Left), Forward, Shoot, Grab, Release, Climb

- Best action is derived from the KB using the ASKVARS query.

  ASKVARS(∃a BestAction(a, 5))

- This returns a binding list that indicates the best action:

  {a / Grab}

The wumpus world

- We add details about the world with simple rules that are tied to time stamps:
- $\forall$ t, s, g, m, c Percept([s, Breeze, g, m, c], t) $\Rightarrow$ Breeze(t).
- $\forall$ t, s, g, m, c Percept([s, None, g, m, c], t) $\Rightarrow$ $\neg$Breeze(t).
- $\forall$ t, s, b, m, c Percept([s, b, Glitter, m, c], t) $\Rightarrow$ Glitter(t).
- $\forall$ t, s, b, m, c Percept([s, b, None, m, c], t) $\Rightarrow$ $\neg$Glitter(t).
- Notice the quantification over time.

The wumpus world

- We add details about the world with simple rules that are tied to time stamps:
- $\forall$ t, s, g, m, c Percept([s, Breeze, g, m, c], t) $\Rightarrow$ Breeze(t).
- $\forall$ t, s, g, m, c Percept([s, None, g, m, c], t) $\Rightarrow$ $\neg$Breeze(t).
- $\forall$ t, s, b, m, c Percept([s, b, Glitter, m, c], t) $\Rightarrow$ Glitter(t).
- $\forall$ t, s, b, m, c Percept([s, b, None, m, c], t) $\Rightarrow$ $\neg$Glitter(t).
- Notice the quantification over time.
- Reflexive behavior:

  $\forall$ t Glitter(t) $\Rightarrow$ BestAction(Grab, t).

ARTIFICIAL INTELLIGENCE

The wumpus world

- Now we need to represent the environment.
- Objects in the environment are:
    - Squares
    - Pits
    - Wumpus
- We could name each square as $square_{12}$ but then we need extra facts to describe the world…
- Instead we use a complex term

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
INSTITUTIONAL OFFICE

The wumpus world

- $\forall$ x,y,a,b Adjacent([x,y],[a,b]) $\Leftrightarrow$ [a,b] $\in$ {[x+1,y],[x-1,y],[x,y+1],[x,y-1]}
- x,y is adjacent to a,b if and only if a,b is in the set of above, below, left or right of x,y.

The wumpus world

- $\forall$ x,y,a,b Adjacent([x,y],[a,b]) $\Leftrightarrow$ [a,b] $\in$ {[x+1,y],[x-1,y],[x,y+1],[x,y-1]}
- At(Agent, s, t)
- We mark an agent on a specific square a at a particular time stamp as this changes over time.

The wumpus world

- $\forall$ x,y,a,b Adjacent([x,y],[a,b]) $\Leftrightarrow$ [a,b] $\in$ {[x+1,y],[x-1,y],[x,y+1],[x,y-1]}
- At(Agent, s, t)
- $\forall$t At(Wumpus,[2,2],t)
- The wumpus stays on a fixed square for all time, but we use the same predicate for indicating this.

The wumpus world

- $\forall$ x,y,a,b Adjacent([x,y],[a,b]) $\Leftrightarrow$ [a,b] $\in$ {[x+1,y],[x-1,y],[x,y+1],[x,y-1]}
- At(Agent, s, t)
- $\forall$t At(Wumpus,[2,2],t)
- $\forall$x, s1, s2, t At(x, s1, t) $\wedge$ At(x, s2, t) $\Rightarrow$ s1 = s2
- We make a rule that objects can only be in one place at a time

The wumpus world

- $\forall$ x,y,a,b Adjacent([x,y],[a,b]) $\Leftrightarrow$ [a,b] $\in$ {[x+1,y],[x-1,y],[x,y+1],[x,y-1]}
- At(Agent, s, t)
- $\forall$t At(Wumpus,[2,2],t)
- $\forall$x, s1, s2, t At(x, s1, t) $\wedge$ At(x, s2, t) $\Rightarrow$ s1 = s2
- $\forall$ s,t At(Agent, s, t) $\wedge$ Breeze(t) $\Rightarrow$ Breezy(s)
- The agent can infer knowledge about the environment and mark squares.

The wumpus world

- $\forall s$ Breezy(s) $\Leftrightarrow \exists r$ Adjacent(r,s) $\wedge$ Pit(r)
- And with the knowledge about which squares are breezy, the agent can infer that there exists at least one pit that is adjacent to each breezy square!

Knowledge engineering process

- General process to construct knowledge base
- Knowledge engineer
- Knowledge engineering process
  - Create general purpose knowledge base for a specific domain with known series of queries

Knowledge engineering process

1. Indentify the questions
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the problem instance
6. Pose queries to the inference procedure and get answers
7. Debug and evaluate the knowledge base